Durham University

Durham Research Online

Deposited in DRO:

11 December 2009

Version of attached file:

Accepted Version

Peer-review status of attached file:

Peer-reviewed

Citation for published item:

Long, Q. and Qiu, Z. and Qin, S. (2003) 'The equivalence of statecharts.', in Formal methods and software engineering : 5th International Conference on Formal Engineering Methods, ICFEM 2003, 5-7 November 2003, Singapore ; proceedings. Berlin: Springer , pp. 125-143. Lecture notes in computer science. (2885).

Further information on publisher's website:

http://dx.doi.org/10.1007/978-3-540-39893-6$_9$

Publisher's copyright statement:

The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-540-39893-6$_9$

Additional information:

Use policy

The full-text may be used and/or reproduced, and given to third parties in any format or medium, without prior permission or charge, for personal research or study, educational, or not-for-profit purposes provided that:

- a full bibliographic reference is made to the original source
- a link is made to the metadata record in DRO
- the full-text is not changed in any way

The full-text must not be sold in any format or medium without the formal permission of the copyright holders.

Please consult the full DRO policy for further details.

Durham University Library, Stockton Road, Durham DH1 3LY, United Kingdom
Tel : +44 (0)191 334 3042 | Fax : +44 (0)191 334 2971
https://dro.dur.ac.uk

The Equivalence of Statecharts*

Quan Long[1], Zongyan Qiu[1], and Shengchao Qin[2]

[1] LMAM, Department of Informatics, School of Mathematical Sciences,
Peking University, Beijing, China 100871
lq@is.pku.edu.cn,zyqiu@pku.edu.cn
[2] Singapore-MIT Alliance, National University of Singapore
qinsc@comp.nus.edu.sg

Abstract. This paper proposes a compositional operational semantics for a nontrivial subset of Statecharts and defines an equivalence relation between Statecharts using bisimulation on configurations. An input/response trace model is also investigated at the level of observable behaviour.

1 Introduction

Statecharts is a visual synchronous specification language introduced by David Harel originally in the early 1980s [4], which is an extension of the finite state machine by hierarchy, concurrency and broadcasting communication. Quoting the words of D.Harel [4],

Statecharts = state diagram + depth +
orthogonality + broadcast communication

Statecharts was invented originally for the development of the avionics system for an Israeli aircraft, and has seen widespread use since then (e.g. [11]). It is desirable to be a tool for specifying real-time, reactive and embedded systems. Some development environments, such as STATEMATE [4, 5], are developed to support the specification of applications with Statecharts. The formalism acts now also as one of the major components of UML [2].

Statecharts can be thought as an enrichment of finite-state transition systems. Here the states can have hierarchical structures and may consist of several sub-states, in fact, sub-Statecharts. These sub-Statecharts can themselves have embedded sub-Statecharts too. Statecharts may be composed sequentially or in parallel to form *Or-Statecharts* or *And-Statecharts* respectively.

The execution of Statecharts is defined by the active states and transitions. In a Statechart, there are usually several simultaneously active states (sub-Statecharts) at a time instant, they communicate with each other via broadcasting events in a global environment. The transitions defined determine the transference of active states. Each of the transitions is labeled by a pair of sets of

* Supported by National Natural Science Foundation of China (No. 60173003)

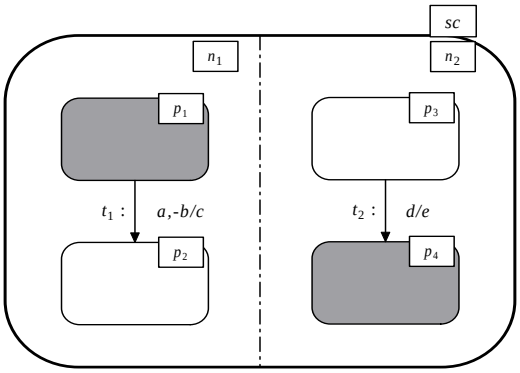


Fig. 1. A simple Statechart

events, where the first set of events is called the *trigger* of the transition which may include both positive and negative events, and the second is referred to as the *action* which can in turn act as triggers to fire other transitions. A transition connects a pair of states, with the first one as its source state and the second one as its target. Intuitively, if the source state of a transition is active, and all positive events from its trigger are present while all negative ones are absent, the transition is enabled and may be performed. When a transition is performed, the events in its action will be generated, and the target state of the transition becomes active afterward.

Fig.1 shows a simple Statechart. It consists of only one *And-Statechart* named *sc*. The Statechart *sc* is composed of two parallel sub-Statecharts named n_1 and n_2. Both n_1 and n_2 are *Or-Statecharts*. Or-Statechart n_1 is refined to basic Statecharts p_1 and p_2, which are connected by transition t_1 with trigger $\{a, -b\}$ (Here we use $-b$ to indicate that event b is absent.) and action c. The figure shows that, in the current situation, the active state of n_1 is p_1. When event a occurs but b does not, t_1 can be performed, thus the event c is generated as the action of t_1, and the active state will be transferred to p_2. On the other side, Or-Statechart n_2 is composed of two basic Statecharts p_3 and p_4, which are connected by transition t_2. The active state of n_2 is p_4.

In literatures, there exist a number of different semantics for variants of Statecharts. M. von der Beeck discussed about twenty variants of Statecharts in [1], each of these variants can be regarded as a subset of the originally proposed language. The version discussed in [5] for STATEMATE has a powerful semantics. But the semantics defined in that paper is neither formal, nor compositional. The work presented in [12] gives a compositional semantics of Statecharts, whereas their version does not contain data states. In [16], M.Schettini, A.Peron and S.Tini had a discussion about the equivalence of Statecharts. They presented a compositional semantics of Statecharts based on Labelled Transition Systems(LTSs). They considered a hierarchy of LTSs equivalences and gave the congruences to Statecharts operators.

In this paper, we present a semantics for Statecharts which is quite similar to that proposed by Qin and Chin [15]. Both semantics have such features as being compositional, adopting an asynchronous time model, reflecting the causality of events, obeying local consistency and covering the data states. The only difference between ours and Qin and Chin's is that in our semantics an active event can be used many times within a macro-step. Based on this semantics, we build a bisimulation between configurations of Statecharts. Borrowing the idea of [9] and [7], we give a definition of the equivalence of Statecharts different from that in [16]. It looks as if our definition were weaker. But actually, from this new definition, we can get the same results as part of that in [16] in a much simpler way. We will have a detailed discussion about this in Section 6. We also have a brief discussion about the traces of Statecharts configurations on the macro-step level.

The next section gives a brief description of term-based syntax of Statecharts. In Section 3, we present our new semantics by a set of operational transition rules. In Section 4, we define the equivalence of Statecharts and prove that the definition is appropriate for getting the needed properties of equivalence. Discussed in Section 5 are the definition and properties of traces of Statecharts configurations. The related works are discussed in Section 6. Finally, Section 7 contains our conclusions and directions for future research.

2 Term-based Syntax of Statecharts

To facilitate our discussion, we use the textual representation for Statecharts that was also given in Qin and Chin [15]. The formal term-based syntax definition for Statecharts is depicted in what follows.

Suppose we have the following sets:

- N: The set of names used to denote Statecharts. We assume that the set is large enough for all the Statecharts.
- Π_e: The set of all positive events.
- $\Pi_{\bar{e}}$: The set of all negative events, that is, $\Pi_{\bar{e}} =_{def} \{\bar{e} | e \in \Pi_e\}$. We assume that $\bar{\bar{e}} = e$.
- Π_a: The set of all assignment actions. These actions have the form $\nu = exp$.
- $\mathcal{T}$: The set of all possible transitions, $\mathcal{T} \subseteq N \times 2^{\Pi_e \cup \Pi_{\bar{e}}} \times 2^{\Pi_e \cup \Pi_a} \times N$, where the first N denotes the source sub-state, the last N is the target sub-state, $2^{\Pi_e \cup \bar{\Pi}_e}$ the trigger and $2^{\Pi_e \cup \Pi_a}$ the new events and assignments which were generated and performed by the transition of $\mathcal{T}$.

Definition 2.1. The set SC of Statecharts is defined inductively as follows:

1. Basic: $N \to SC$:

   $$Basic(n) =_{def} [n]$$

2. Or: $N \times \langle SC \rangle \times SC \times 2^{\mathcal{T}} \to SC$:

   $$Or(n, \langle P_1, \cdots, P_l, \cdots, P_m \rangle, P_l, T) =_{def} [n, (P_1, \cdots, P_m), P_l, T]$$

3. And: $N \times 2^{SC} \rightarrow SC$:

$$And(n, \{P_1, \cdots, P_m\}) =_{def} [n, (P_1, \cdots, P_m)]$$

Note that we use square brackets to enclose a Statechart, use $\langle SC \rangle$ to denote all sequences of Statecharts of SC. Following are some explanations of the constructions of Statecharts.

- $Basic(n)$ denotes a basic Statechart named n.
- $Or(n, \langle P_1, \cdots, P_l, \cdots, P_m \rangle, P_l, T)$ denotes an Or-Statechart named n with a sequence of sub-states $\langle P_1, \cdots, P_m \rangle$, where P_1 is the default sub-state and P_l is the active sub-state currently. Notice that the sub-states are defined as a sequence rather than a set, to indicate that P_1 is the default sub-state. The order of other sub-states is arbitrary. T is the set composed of all possible transitions among the sub-states of n.
- $And(n, \{P_1, \cdots, P_m\})$ denotes the And-Statechart named n, which contains a number of parallel sub-states $P_1, \cdots, P_m$, where $P_1, \cdots, P_m$ are basic Statecharts or Or-Statecharts (but not And-Statecharts).

Example 2.1. The term-based syntax for the Statechart shown in **Fig.**1 is given below:

1. $N = And(sc, \{n_1, n_2\}) = [sc, (n_1, n_2)]$;
2. $n_1 = Or(n_1, \langle p_1, p_2 \rangle, p_1, \{t_1\}) = [n_1, (p_1, p_2), p_1, \{\langle p_1, \{a, \bar{b}\}, \{c\}, p_2 \rangle\}]$;
3. $n_2 = [n_2, (p_3, p_4), p_4, \{\langle p_3, \{d\}, \{e\}, p_4 \rangle\}]$;
4. Definition of p_1, p_2, p_3, p_4, etc.

Note that we use $\langle p_i, E, A, p_j \rangle$ to represent a transition from state p_i to p_j with trigger set E and action set A. □

It should be noticed that our version is a subset of Harel's original definition. We do not include timeout events, inter-level transitions and some other minor features.

3 Operational Transition Rules

Before presenting the semantics for Statecharts, we define configurations of Statecharts first. A configuration of Statecharts is defined as a triple $\langle P, \nu, E \rangle$, where

- P is the syntax of the Statechart of interest.
- ν is a snapshot of data items (data state).
- $E \subseteq \Pi_e$ is a set of active events.

The behavior of a Statechart is composed of a sequence of macro-steps, each of which comprises a sequence of micro-steps which are triggered by the external or internal events. A Statechart reacts to any stimulus from the environment at the beginning of each macro-step by performing a sequence of transitions and generating some internal events (by the actions of the transitions it performs),

which can in turn fire other state transitions and lead to a chain of micro-steps without advancing time. During this chain of micro-steps, the Statechart does not respond to any (potentially) further external stimulus. In case that no more transitions, except for the clock tick, are enabled, the macro-step comes to the end. The clock tick transition then occurs, which empties the set of currently active events and advances time by one unit. Then, the Statechart is ready again to accept another external stimuli and start off the next macro-step. The relationship of macro-step and micro-step was discussed in details by G. Lüttgen, M. von der Beeck and R. Cleaveland [12].

We explore the following transition rules, consisting of state transitions rules and time advance transitions rules.

The first transition rule initiates a macro-step for a Statechart. It is the first micro-step of a macro-step. It performs only when a set of events E arrives (due to the environment) and the Statechart is ready to accept them.

Rule 3.1 (*Initiate*). $\langle P, \nu, \phi \rangle \xrightarrow{E} \langle P, \nu, E \rangle$ □

In an Or-Statechart, if a transition between two immediate connected sub-states is enabled, the transition can be performed.

Rule 3.2 (*Or*). Suppose P is an Or-Statechart and $P = [n, (P_1, \cdots, P_m), P_l, T]$, $\tau \in En(P, E)$. Then we can have

$$\langle P, \nu, E \rangle \xrightarrow{\tau} \langle [n, (P_1, \cdots, P_m), a2d(tgt(\tau)), T], \nu', E \cup act^e(\tau) \rangle$$

where

- $En(P, E) =_{def} \{\tau \in T | sre(\tau) = P_l \wedge trig^+(\tau) \subseteq E \wedge trig^-(\tau) \cap E = \phi\}$ is the set of transitions enabled in current configuration on the "highest level".
- $sre(\tau)$ and $tgt(\tau)$ are the source and target states of transition τ, respectively.
- $act^e(\tau)$ denotes the set of events generated by transition τ.
- $trig^+(\tau)$ and $trig^-(\tau)$ are respectively the set of positive events and the set of negative ones that form the trigger of the transition τ.
- The function $a2d(P)$ maps the sub-state p of P to its default sub-state (recursively). Its definition is:

  $a2d([n]) =_{def} [n]$
  $a2d([n, (P_1, \cdots, P_m), P_l, T]) =_{def} [n, (P_1, \cdots, P_m), a2d(P_1), T]$
  $a2d([n, (P_1, \cdots, P_m)]) =_{def} [n, (a2d(P_1), \cdots, a2d(P_m))]$

- ν' denotes the new data states which might be updated by actions of τ. □

If no transition among immediate sub-states of an Or-Statechart is enabled, then the transitions in its active sub-state can be performed.

Rule 3.3 (*Or-Substate*). Suppose $P = [n, (P_1, \cdots, P_m), P_l, T]$ is an Or-Statechart, $En(P, E) = \phi$, and $\langle P_l, \nu, E \rangle \xrightarrow{\tau} \langle P_l', \nu', E' \rangle$, then

$$\langle P, \nu, E \rangle \xrightarrow{\tau} \langle [n, (P_1, \cdots, P_m), P_l', T], \nu', E' \rangle$$

□

From **Rule 3.3** we know that, the enabled transitions of the higher level Statechart will have the relative higher priority of being chosen, while simultaneously enabled transitions of the embedded Statecharts will be discarded.

Notice that the transition τ in above rule may be the conjunction of a set of transitions, because P_l can be an And-Statechart (See **Rule 3.4** below). We use also symbol τ to denote that case for convenience and shall follow this convention when needed. On the other hand, the fired transition(s) τ is (are) definitely on the highest possible level in P_l due to **Rule 3.2** and **Rule 3.3**.

If each variable can be modified by only one transition of an And-Statechart, then all enabled transitions of those sub-states can perform together. We use $WV(\tau_i)$ to denotes the variables that can be modified by τ_i. Here we avoid the racing conflicts only for simpleness. Adding it will not bring essential changes to the main parts of this paper.

Rule 3.4 (*And*). Suppose P is an And-Statechart, $P = [n, (P_1, \cdots, P_m)]$. For $i = 1, 2, \cdots, m$, P_i is a Basic Statechart or Or-Statechart,

$$\langle P_i, \nu, E\rangle \xrightarrow{\tau_i} \langle P_i', \nu_i', E \cup act^e(\tau_i)\rangle$$

If $En^*(P_i, E) = \phi$ for some i, then the sub-configuration is considered as staying the same. That is

$$\langle P_i, \nu, E\rangle \rightarrow \langle P_i, \nu, E\rangle$$

where En^* is defined as follows

$$\begin{array}{l} En^*([n], E) =_{def} \phi \\ En^*(P = [n, (P_1, \cdots, P_m), P_l, T], E) =_{def} En(P, E) \cup En^*(P_l, E) \\ En^*(P = [n, (P_1, \cdots, P_m)], E) =_{def} \bigcup_{i=1}^{m} En^*(P_i, E) \end{array}$$

We have further condition that for all $i \neq j$, $WV(\tau_i) \cap WV(\tau_j) = \phi$, we denote $\nu' = \bigoplus_{i=1}^{m} \nu_i'$ the direct sum of all ν_i', then we have

$$\langle P, \nu, E\rangle \xrightarrow{\bigwedge_{i=1}^{m} \tau_i} \langle [P, (P_1', \cdots, P_m')], \nu', E \cup \bigcup_{i=1}^{m} act^e(\tau_i)\rangle$$

□

If no transition is enabled in a Statechart and all of its embedded sub-states, the current macro-step comes to the end. The Statechart will clear the set of events and advance the time (Here σ is used to denote the clock tick transition), and is ready to perform **Rule 3.1** to start another macro-step.

Rule 3.5 (*Empty and Time Advance*). If $En^*(P, E) = \phi$, then we have

$$\langle P, \nu, E\rangle \xrightarrow{\sigma} \langle P, \nu, \phi\rangle$$

□

Here is a simple example of how these operational rules work.

Example 3.1. In the Statechart of **Fig.1**, the default configuration is

$$\langle [sc, ([n_1, (p_1, p_2), p_1, t_1], [n_2, (p_3, p_4), p_3, t_2])], \nu, \phi \rangle$$

When the external events set $\{a, d\}$ appears. **Rule 3.1** works.

$$\begin{aligned} &\langle [sc, ([n_1, (p_1, p_2), p_1, t_1], [n_2, (p_3, p_4), p_3, t_2])], \nu, \phi \rangle \\ \xrightarrow{\{a,d\}} &\langle [sc, ([n_1, (p_1, p_2), p_1, t_1], [n_2, (p_3, p_4), p_3, t_2])], \nu, \{a, d\} \rangle \end{aligned}$$

This is an And-Statechart. Following the **Rule 3.4**, we need to consider its sub-Statecharts. According to **Rule 3.2**, the following two potential transitions are ready to be fired:

$$\begin{aligned} &\langle [n_1, (p_1, p_2), p_1, t_1], \nu_1, \{a, d\} \rangle \xrightarrow{t_1} \langle [n_1, (p_1, p_2), p_2, t_1], \nu_1', \{a, d, c\} \rangle \\ &\langle [n_2, (p_3, p_4), p_3, t_2], \nu_2, \{a, d\} \rangle \xrightarrow{t_2} \langle [n_2, (p_3, p_4), p_4, t_2], \nu_2', \{a, d, e\} \rangle \end{aligned}$$

The conditions of **Rule 3.4** hold. Therefore,

$$\begin{aligned} &\langle [sc, ([n_1, (p_1, p_2), p_1, t_1], [n_2, (p_3, p_4), p_3, t_2])], \nu, \{a, d\} \rangle \\ \xrightarrow{t_1 \wedge t_2} &\langle [sc, ([n_1, (p_1, p_2), p_2, t_1], [n_2, (p_3, p_4), p_4, t_2])], \nu', \{a, d, c, e\} \rangle \end{aligned}$$

where $\nu' = \nu_1' \oplus \nu_2'$.

Now the set $En^*(sc, \{a, c, d, e\})$ is empty, hence the **Rule 3.5**, that is,

$$\begin{aligned} &\langle [sc, ([n_1, (p_1, p_2), p_2, t_1], [n_2, (p_3, p_4), p_4, t_2])], \nu', \{a, d, c, e\} \rangle \\ \xrightarrow{\sigma} &\langle [sc, ([n_1, (p_1, p_2), p_2, t_1], [n_2, (p_3, p_4), p_4, t_2])], \nu', \phi \rangle \end{aligned}$$

A macro-step comes to the end. □

4 Equivalence

For the sake of convenience, we will use the capital letter C (or C_i) to denote the configuration $\langle P, \nu, E \rangle$ and let $\mathbb{C}$ be the space of all possible configurations of a set of Statecharts.

For the description of the set of events used in one micro-step to trigger the transition, we give the following definition.

Definition 4.1. We use $C \xrightarrow{E} C'$ to denote that, the configuration C evolves to C' in one **micro-step** by some fired transitions (There might be more than one fired transitions because of **Rule 3.4**), and E is the set of all the positive events of the triggers of all the transitions performed in this micro-step.

The following definition describes the configurations which will execute micro-steps infinitely and, therefore, makes the Statecharts no chance to participate further stimuli from the environment.

Definition 4.2 (*Divergent*). A configuration C is divergent if there is an infinite sequence of configurations $\{C_n\}_{n=1}^{\infty}$ such that $C = C_1$ and $C_n \xrightarrow{E_n} C_{n+1}$, where E_n is the corresponding set of events. That is, there is an infinite sequence of micro-steps started from C.

In what follows we depict the relationship of two configurations to express their equivalent property.

Definition 4.3 (*Bisimulation*). A binary relation $\mathcal{S}$ over a configuration space $\mathbb{C}$ is a bisimulation iff it satisfies the following conditions:

1. $\mathcal{S}$ is an equivalence relation.
2. Given $C_i = \langle P_i, \nu_i, E_i \rangle, i = 1, 2$. If $C_1 \mathcal{S} C_2$ then
   (a) $var(P_1) = var(P_2)$, where $var()$ denotes the variable set
   (b) $\nu_1 = \nu_2$
   (c) $E_1 = E_2$
   (d) if C_1 is not divergent, For any set of events E, whenever there exists a C_1' such that $C_1 \xrightarrow{E} C_1'$, then there exists C_2', such that
   $$(C_2 \xrightarrow{E} C_2') \wedge (C_1' \mathcal{S} C_2')$$
   (e) If $C_1 \xrightarrow{\sigma} C_1'$ (**Rule 3.5**), then there exists C_2', such that
   $$(C_2 \xrightarrow{\sigma} C_2') \wedge (C_1' \mathcal{S} C_2')$$

□

In this definition, we do not mention the actions of the performed transitions. However, from 2 (b), (c) and (d) we know that, the data states and sets of active events of C_1' and C_2', i.e. ν_1' and ν_2', E_1' and E_2' (which reflect the effects of the actions of the micro-step), are the same.

The following lemma shows our definition of bisimulation preserves normal operations.

Lemma 4.4. If $\{\mathcal{S}_i\}$ are bisimulations, then the following relations are also bisimulations.

1. $\bigcup_i \mathcal{S}_i$
2. $\mathcal{S}_i \circ \mathcal{S}_j$

Proof. The proof of 1 and 2 are similar. We prove 2 as an example.

What needs to be checked are the two conditions of **Definition 4.3** in turn. The condition 1 and (a), (b) and (c) of condition 2 are trivial, let's see the condition 2 (d).

If $C_1(\mathcal{S}_i \circ \mathcal{S}_j)C_3$, then there exists a configuration C_2 such that

$$(C_1 \mathcal{S}_i C_2) \wedge (C_2 \mathcal{S}_j C_3)$$

So if there exists a configuration C_1' such that $C_1 \xrightarrow{E} C_1'$, then

$$\exists C_2' \cdot (C_2 \xrightarrow{E} C_2') \wedge (C_1' \mathcal{S}_i C_2')$$

For $C_2 \mathcal{S}_j C_3$ and $C_2 \xrightarrow{E} C_2'$, we have

$$\exists C_3' \cdot (C_3 \xrightarrow{E} C_3') \wedge (C_2' \mathcal{S}_j C_3')$$

So we have $C_1'(\mathcal{S}_i \circ \mathcal{S}_j)C_3'$. Now we have proved that $C_1 \xrightarrow{E} C_1'$ implies

$$\exists C_3' \cdot (C_3 \xrightarrow{E} C_3') \wedge (C_1'(\mathcal{S}_i \circ \mathcal{S}_j)C_3')$$

In case of condition (e), it is similar to condition (d). □

Using **definition 4.3**, we give the definition of the equivalence of two configurations as follows.

Definition 4.5 (*Configuration Equivalence*). Two configurations C_1 and C_2 are equivalent, denoted by $C_1 \sim C_2$, iff there exists a bisimulation $\mathcal{S}$ such that $C_1 \mathcal{S} C_2$.

Furthermore, we give the definition for the equivalence of two Statecharts as follows. It seems that this definition is a little weak. In fact, this definition is sufficient. The validity of this statement will be embodied by later theorems and corollaries.

Definition 4.6 (*Statechart Equivalence*). Two Statecharts P and Q are equivalent, denoted by $P \sim Q$, iff $D(P) \sim D(Q)$, where $D(P)$ denotes the default configuration of P. That is, the configuration where the set of active states is exactly the set of default states of P.

Example 4.1. $[sc_1, ([n_{11}, (p_1, p_2), p_1, t], [n_{12}, (p_3, p_4), p_3, t])]$ and $[sc_2, ([n_{21}, (p_1, p_3)], [n_{22}, (p_2, p_4)]), n_{21}, t]$ are two Statecharts, where p_1, p_2, p_3, p_4 are their embedded Statecharts. Assuming $var(sc_1) = var(sc_2)$, $\nu(D(sc_1)) = \nu(D(sc_2))$ and $E(D(sc_1)) = E(D(sc_2))$, these two Statecharts are equivalent.

Fig.2 and **Fig.3** shows these two Statecharts. The only micro-step can be fired of sc_1 is to transfer p_1, p_3 to p_2, p_4 parallelly by **Rule 3.4** and the only micro-step can be fired of sc_2 is to transfer n_{21} to n_{22} by **Rule 3.2**. It is easy to check that their default configurations can bisimulate each other. □

Lemma 4.7. $\sim$ on the Statechart space is an equivalence relation.
The proof of this lemma is trivial. □

To illustrate the validity of our definition of the equivalence, we shall show the result that for every possible configuration of a Statechart P, we can find a configuration from a Statechart Q which is equivalent to P, these configurations bisimulate each other. We give the following definition first.

Definition 4.8. Suppose $tr = \langle E_1, E_2, \cdots \rangle$, where $E_i \subseteq \Pi_e$ (Recall that Π_e is the set of all possible events) or $E_i = \{\sigma\}$ is a sequence of sets of events. We use $C_1 \xrightarrow{tr} C_2$ to denote the fact that the configuration C_1 evolves into C_2 by performing micro-steps $\langle step_1, step_2, \cdots \rangle$ in turn and the set of positive events out of triggers of the transitions fired in $step_i$ is E_i.

Given two equivalent Statecharts P and Q ($P \sim Q$), the following theorem states that, for any reachable configuration in a run of P (or Q respectively), there exists a bisimular configuration in a run of Q (or P respectively).

Theorem 4.9. Suppose $P \sim Q$. Let tr be any finite length sequence of sets of events. If there exists a configuration C_p such that $D(P) \xrightarrow{tr} C_p$, then there exists a configuration C_q such that $D(Q) \xrightarrow{tr} C_q$ and $C_p \sim C_q$.

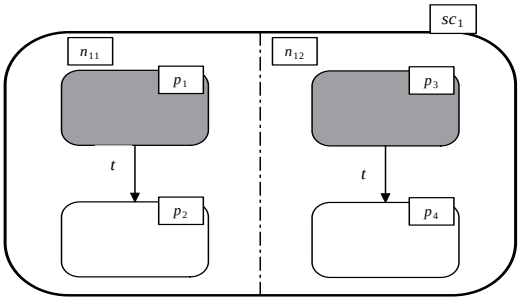


Fig. 2. Statechart sc_1

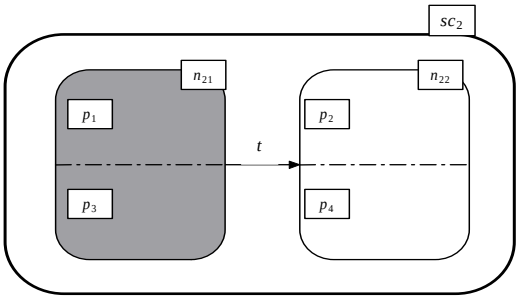


Fig. 3. Statechart sc_2

That is, we have the following commuting diagram:

$$\begin{array}{ccc} D(P) & \xrightarrow{tr} & C_p \\ \sim\downarrow & & \downarrow\sim \\ D(Q) & \xrightarrow[tr]{} & C_q \end{array}$$

Proof. By induction on n, the length of tr.
(1) $n = 1$. By **definition 4.5** and **definition 4.6**, there exists a bisimulation $\mathcal{S}$ such that

$$D(P)\,\mathcal{S}\,D(Q),$$

and we have

$$D(P) \xrightarrow{E_1} C_p$$

From 2.(d) and 2.(e) in **Definition 4.3**, we get that there exists C_q such that

$$(D(Q) \xrightarrow{E_1} C_q) \wedge (C_p \mathcal{S} C_q)$$

(2) Assume the result holds for $n = k$. We prove that it also holds for the case of $n = k + 1$.

Suppose $tr = \{E_1, E_2, \cdots, E_{k+1}\}$. We denote $tr' = \{E_1, E_2, \cdots, E_k\}$. Then there exists C'_p such that $D(P) \xrightarrow{tr'} C'_p$ and $C'_p \xrightarrow{E_{k+1}} C_p$. Using the inductive assumption, there exists a configuration C'_q such that $D(Q) \xrightarrow{tr'} C'_q$ and $C'_p \sim C'_q$. By **Definition 4.5** and **Definition 4.6**, there exists a bisimulation $\mathcal{S}$ such that

$$(C'_p \mathcal{S} C'_q) \wedge (C'_p \xrightarrow{E_{k+1}} C_p)$$

From 2.(d) and 2.(e) in **Definition 4.3**, we know that there exists C_q such that

$$(C'_q \xrightarrow{E_{k+1}} C_q) \wedge (C_p \mathcal{S} C_q)$$

That is

$$(D(Q) \xrightarrow{tr} C_q) \wedge (C_p \mathcal{S} C_q)$$

Now with (1) and (2) done, we have come to the end of our proof. □

From the above theorem we can prove the following property which expresses the above mentioned idea easily.

Corollary 4.10. Suppose $P \sim Q$. Then for each legal configuration C_p of P, there exists a configuration C_q of Q such that $C_p \sim C_q$.

Proof. Consider the micro-step sequence $\langle step_1, \cdots, step_k \rangle$ which leads $D(P)$ to C_p and the corresponding sequence of sets of events $\langle E_1, \cdots, E_k \rangle$. □

The following theorem shows that the equivalence relation is preserved by the constructors of Statecharts.

Theorem 4.11 (*Congruence*). $P_i \sim Q_i$ $(i = 1, \cdots, m)$ implies

1. $And(N_p, \{P_1, \cdots, P_m\}) \sim And(N_q, \{Q_1, \cdots, Q_m\})$;
2. $Or(N_p, \langle P_1, \cdots, P_m \rangle, P_l, T) \sim Or(N_q, \langle Q_1, \cdots, Q_m \rangle, Q_l, T')$. where there exist a bijection f between T and T', such that for any $\tau \in T$
   (a) $act^e(\tau) = act^e(f(\tau))$
   (b) $trig^+(\tau) = trig^+(f(\tau)) \wedge trig^-(\tau) = trig^-(f(\tau))$
   (c) $P_i \xrightarrow{\tau} P_j \Longleftrightarrow Q_i \xrightarrow{f(\tau)} Q_j$

That is, we have following commuting diagram in which the symbol *op* denotes the Statechart construction operators *And* or *Or*:

$$\begin{array}{ccc} \{P_1, \cdots, P_m\} & \xrightarrow{op} & N_p \\ \sim\downarrow & & \downarrow\sim \\ \{Q_1, \cdots, Q_m\} & \xrightarrow[op]{} & N_q \end{array}$$

Proof. 1. For $P_i \sim Q_i$, we have $D(P_i) \sim D(Q_i)$, then $\exists \mathcal{S}_i \cdot D(P_i)\mathcal{S}_i D(Q_i)$, where $\mathcal{S}_i$ is a bisimulation. So we have:
For any set of events E, whenever there exists a configuration $(D(P_i))'$ such that

$$D(P_i) \xrightarrow{E} (D(P_i))',$$

there exists $(D(Q_i))'$, such that

$$(D(Q_i) \xrightarrow{E} (D(Q_i))') \wedge ((D(P_i))' \mathcal{S}_i (D(Q_i))')$$

Now we define the relationship

$$\mathcal{S} = \{\langle D(And(N_p, \{P_1, \cdots, P_m\})), D(And(N_q, \{Q_1, \cdots, Q_m\}))\rangle \\ \text{where } D(P_i)\mathcal{S}_i D(Q_i)\} \quad \cup Id$$

We prove that $\mathcal{S}$ is a bisimulation as follows.
It is trivial that $\mathcal{S}$ is an equivalence relation. (†)
Now we check the conditions 2(a) – 2(e) in **Definition 4.3**. 2(a), 2(b) and 2(c) are trivial. Since 2(e) is similar to 2(d), we check condition 2(d) in details here.
Since our discussion is at the micro-step level, the actions of transitions in $D(P_i)$ do not have effect on $D(P_j)$.
Thus we have the following fact.

$$\text{For any set of events } E, \text{ whenever } D(N_p) \xrightarrow{E} (D(N_p))',$$

there exists $(D(N_q))'$, such that

$$(D(N_q) \xrightarrow{E} (D(N_q))') \wedge ((D(N_p))' \mathcal{S} (D(N_q))') \quad (\ddagger)$$

We then have $N_p \sim N_q$ from (†) and (‡).

2. Similar to 1, we have $\exists \mathcal{S}_i \cdot D(P_i)\mathcal{S}_i D(Q_i)$, where $\mathcal{S}_i$ is a bisimulation, and the following result:
For any set of events E, whenever there exists a configuration $(D(P_i))'$ such that

$$D(P_i) \xrightarrow{E} (D(P_i))',$$

there exists a $(D(Q_i))'$, such that

$$(D(Q_i) \xrightarrow{E} (D(Q_i))') \wedge ((D(P_i))' \mathcal{S}_i (D(Q_i))')$$

Now we define the relationship

$$\mathcal{S} = \{\langle D(Or(N_p, \langle P_1, \cdots, P_m\rangle, P_l, T)), \\ D(Or(N_q, \langle Q_1, \cdots, Q_m\rangle, Q_l, T'))\rangle, \text{ where } D(P_i)\mathcal{S}_i D(Q_i)\} \\ \cup \quad Id$$

It is trivial that $\mathcal{S}$ is an equivalence relation. $(*)$

Analogically, to check condition 2 in **Definition 4.3**, we need a formula similar to (‡). It can be divided into two cases:
(a) If the micro-step triggered by E is between the immediate sub-state P_l and P_k, i.e.

$$D(N_p) \xrightarrow{E} (D(N_p))' = D(Or(N'_p, \langle P_1, \cdots, P_m \rangle, P_k, T))$$

Because of there is a bijection f between the transitions sets of N_p and N_q which satisfies the three conditions, we have

$$D(N_q) \xrightarrow{E} (D(N_q))' = D(Or(N'_q, \langle Q_1, \cdots, Q_m \rangle, Q_k, T'))$$

(b) If the micro-step triggered by E is in the active sub-state P_l, i.e.

$$P_l \xrightarrow{E} P'_l$$

For $P_l \mathcal{S}_l Q_l$, we have there exits Q'_l, such that

$$(Q_l \xrightarrow{E} Q'_l) \wedge (P'_l \mathcal{S}_l Q'_l)$$

We take

$$(D(N_q))' = D(Or(N'_q, \langle Q_1, \cdots, Q_m \rangle, Q'_l, T'))$$

In both case (a) and case (b) it is trivial that

$$(D(N_q) \xrightarrow{E} (D(N_q))') \wedge ((D(N_p))' \mathcal{S} (D(N_q))') \qquad (**)$$

From $(*)$ and $(**)$, we obtain $N_p \sim N_q$. □

5 Traces

As shown in the **Definition 4.8**, suppose $tr = \langle E_1, E_2, \cdots \rangle$ is a sequence of sets of events. We use $C_1 \xrightarrow{tr} C_2$ to denote the fact that the configuration C_1 evolves into C_2 by performing micro-steps $\langle step_1, step_2, \cdots \rangle$ in turn and the set of positive events from the triggers of the transitions fired in $step_i$ is E_i. In this section we investigate some properties on the level of **macro-step**.

Definition 5.1. We use Π_{ex} to denote all possible external events. Suppose $E \subseteq \Pi_{ex}$ and $C_i = \langle P_i, \nu_i, \phi \rangle$, $i = 1, 2$, we use $C_1 \stackrel{E}{\Longrightarrow} C_2$ to denote a macro-step from C_1 to C_2 with the set of initial external events E by a sequence of micro-steps, where only the last micro-step is the clock tick σ.

We use $(2^{\Pi_{ex}})^*$ to denote the set of all the possible finite-length sequences of sets of external events. Suppose $tr = \langle E_1, E_2, \cdots, E_m \rangle \in (2^{\Pi_{ex}})^*$. We use also $C_1 \stackrel{tr}{\Longrightarrow} C_2$ to denote the fact that the configuration C_1 can evolve into C_2 by performing a sequence of macro-steps $\langle Mstep_1, Mstep_2, \cdots, Mstep_m \rangle$ in turn and the set of events E_i is the set of initial events stimulating the $Mstep_i$.

When a finite sequence of sets of external events comes sequentially, a Statechart starts to respond the first set of events from its current configuration. As reactions to this finite sequence of stimuli, it may perform a sequence of transitions which are triggered by these stimuli directly or indirectly, go through a number of macro-steps and reach another configuration eventually, or it may fall into divergence in some macro-step on the way, and is not able to participate the next macro-step. We address these issues in this section.

We give two definitions to formalize the aforementioned ideas.

Definition 5.2 (*Trace*). A trace $tr \in (2^{\Pi_{ex}})^*$ is a finite sequence of external events in which a particular Statechart participates with its environment.

What follows is our definition of specific trace sets. We propose two type of sets with respect to configurations. For a configuration C_P, set $Div(C_P)$ includes all the traces that **may** lead the configuration C_P to divergence, while the another set is $Prg(C_P)$ which includes all traces which definitely lead configuration C_P to a steady configuration. We use Prg to hint that the configuration will progress normally while "consuming" a trace of $Prg(C_P)$ and will be ready to accept other external events.

Definition 5.3 (*Trace Sets*). Suppose P is a Statechart and C_P is one of its configurations with empty set of events, we define two sets of sequences of set of events as follows:

$$\begin{aligned} Div(C_P) =_{def} \{\ & tr \in (2^{\Pi_{ex}})^* | \\ & \exists s, C' \cdot s \prec tr \wedge (C_P \stackrel{s}{\Longrightarrow} C') \wedge div(C', tr(\#s+1))\} \\ Prg(C_P) =_{def} \{\ & tr \in (2^{\Pi_{ex}})^* | \exists C' \cdot (C_P \stackrel{tr}{\Longrightarrow} C')\rangle\} \end{aligned}$$

where $s \prec tr$ means s is a proper prefix of tr. We use $div(C', E)$ to represent that C' is divergent after receiving set of events E according to **Rule 3.1**. Note that $\#s$ is the length of s, while $tr(n)$ denotes the nth element of tr.

Form the definition we have the following property immediately.

Lemma 5.4. Suppose C_P is a configuration of a Statechart P with empty set of events, then

$$Prg(C_P) \cup Div(C_P) = (2^{\Pi_{ex}})^*$$

Proof. By **Definition 4.2**, we have the fact that if a trace tr is not in $Div(C_P)$, then there exists a steady configuration C as the end configuration after preforming all the macro-steps triggered by tr. From the transition **Rule 3.1**$\sim$ **3.5**, we can see that only the configuration with the form $\langle P, \nu, \phi\rangle$ can be the end configuration of a macro-step. Therefore tr lies in $Prg(C_P)$. So we have

$$Prg(C_P) \cup Div(C_P) = (2^{\Pi_{ex}})^*$$

□

Two equivalent configurations of Statecharts should have the same trace sets Div and Prg as one may expect. The following theorem tells us the truth.

Theorem 5.5 (*Trace*). Suppose $P \sim Q$ and C_p is the configuration of P with empty set of events. If C_q is the configuration of Q which is equivalent to C_p. Then

$$Div(C_p) = Div(C_q) \quad \text{and} \quad Prg(C_p) = Prg(C_q)$$

Proof. If $tr \in Prg(C_p)$ is a trace, then there exists a configuration C'_p such that

$$C_p \stackrel{tr}{\Longrightarrow} C'_p$$

Because a macro-step can be considered as a sequence of micro-steps, by **Theorem 4.9**, we see that there exists a configuration C'_q, such that

$$C_q \stackrel{tr}{\Longrightarrow} C'_q$$

So $tr \in Prg(C_q)$, Then we have

$$Prg(C_p) \subseteq Prg(C_q)$$

For the same reason , we have

$$Prg(C_q) \subseteq Prg(C_p)$$

So we come to

$$Prg(C_p) = Prg(C_q)$$

From **Lemma 5.4**, we obtain $Div(C_p) = Div(C_q)$. □

We considered above the *external traces* or *input traces*, which are provided by the environment. Now we take into account the responses of a Statechart to these stimuli from the environment, and introduce the *response traces*.

According to our definition of transition rules, before a clock tick transition, all events generated by those transitions in one macro-step are accumulated in the set of active events. This reflects the reaction of the Statechart to environmental stimuli arrived at the beginning of the macrostep. We use it to specify a Statechart's response behaviour to the environment.

Definition 5.6 (*Response*). We use $C \stackrel{E/\hat{E}}{\Longrightarrow} C'$ to denote a macro-step from C to C' with the set of initial external events E by a sequence of micro-steps, where only the last micro-step is the clock tick σ, and the set of events in the configuration before the clock tick is $\hat{E}$. We call $\hat{E}$ the set of response events in this macro-step.

Definition 5.7 (*Response Trace*). Suppose P is a Statechart and C_p is one of its configuration with empty set of events. Suppose $tr = \langle E_1, E_2, \cdots, E_m \rangle \in (2^{\Pi_{ex}})^*$ and there is a configuration C' such that $C \stackrel{tr}{\Longrightarrow} C'$. We collect the sets of response events along the way, which form a sequence of sets of events $\hat{tr} = \langle \hat{E}_1, \hat{E}_2, \cdots, \hat{E}_m \rangle \in (2^{\Pi})^*$, and call the sequence a *response trace* of tr with respect to C and C', and denote the fact as $C \stackrel{tr/\hat{tr}}{\Longrightarrow} C'$.

Obviously, due to possible non-determinism, for a certain C and a fixed tr, there might be more than one pair of C' and $\hat{tr}$ such that $C \stackrel{tr/\hat{tr}}{\Longrightarrow} C'$.

In the remainder of this section, we shall prove that, if two configurations are equivalent, $C_1 \sim C_2$, or two Statecharts are equivalent, $P \sim Q$, then for an external event trace tr, they will generate the same set of response traces.

Lemma 5.8. Suppose $P \sim Q$, C_p is a configuration of P with empty set of events and C_q is a configuration of Q which is equivalent to C_p. Suppose $E \in 2^{\Pi_{ex}}$ and $\hat{E} \in 2^{\Pi}$. Then we have that, if there exists a configuration C'_p of P such that $C_p \stackrel{E/\hat{E}}{\Longrightarrow} C'_p$, then there exists a configuration C'_q of Q such that $C_q \stackrel{E/\hat{E}}{\Longrightarrow} C'_q$. That is we have the following commuting diagram:

$$\begin{array}{ccc} C_p & \xrightarrow{\stackrel{E/\hat{E}}{\Longrightarrow}} & C'_p \\ \sim\downarrow & & \downarrow\sim \\ C_q & \xrightarrow{\stackrel{E/\hat{E}}{\Longrightarrow}} & C'_q \end{array}$$

Proof. A macro-step can be considered as a sequence of micro-steps, so if we have $\exists C'_p \cdot C_p \stackrel{E/\hat{E}}{\Longrightarrow} C'_p$, then by **Theorem 4.9**, we see that

$$\exists C'_q \cdot C_q \stackrel{E}{\Longrightarrow} C'_q \qquad (*)$$

Suppose the macro-step from C_p to C'_p is $Mstep_p = \langle step_1, \cdots, step_{k-1}, step_k \rangle$, the macro-step from C_q to C'_q is $Mstep_q = \langle step'_1, \cdots, step'_{k-1}, step'_k \rangle$, and the configuration sequences with respect to these steps are $\langle C_{p,1}, \cdots, C_{p,k-1}, C'_p \rangle$ and $\langle C_{q,1}, \cdots, C_{q,k-1}, C'_q \rangle$ respectively, where $step_i$ and $step'_i$ $(i = 1, \cdots, k)$ are all micro-steps. For $\hat{E}$ is the response set of $Mstep_p$, from **Definition 5.6**, we know that the set of events in $C_{p,k-1}$ is $\hat{E}$. For $C_{p,k-1} \sim C_{q,k-1}$, from **Definition 4.3,4.5** we get that the event set of $C_{q,k-1}$ is also $\hat{E}$. So we have $\hat{E}$ is the response set of $Mstep_q$. Then from $(*)$ we have come to

$$\exists C'_q \cdot C_q \stackrel{E/\hat{E}}{\Longrightarrow} C'_q$$

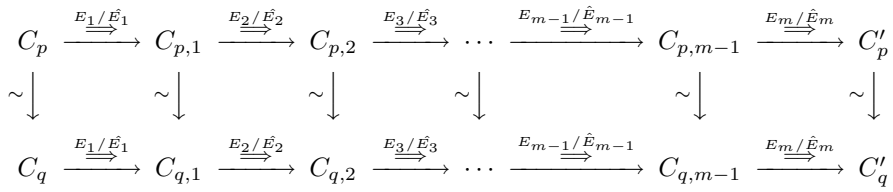
So there is

$$\exists C'_p \cdot C_p \stackrel{E/\hat{E}}{\Longrightarrow} C'_p \text{ implies } \exists C'_q \cdot C_q \stackrel{E/\hat{E}}{\Longrightarrow} C'_q$$

□

Theorem 5.9 (*Response Trace*). Suppose $P \sim Q$, C_p is a configuration of P with empty set of events and C_q is a configuration of Q which is equivalent to C_p. Suppose $tr \in (2^{\Pi_{ex}})^*$ and $\hat{tr} \in (2^{\Pi})^*$ is a sequence with the same length of tr. Then we have that, if there exists a configuration C'_p of P such that $C_p \stackrel{tr/\hat{tr}}{\Longrightarrow} C'_p$, then there exists a configuration C'_q of Q such that $C_q \stackrel{tr/\hat{tr}}{\Longrightarrow} C'_q$.

Proof. Suppose $tr = \langle E_1, E_2, \cdots, E_m \rangle$ and $tr = \langle \hat{E}_1, \hat{E}_2, \cdots, \hat{E}_m \rangle$, By **Lemma 5.8** and the following commuting diagram,



One can see the existence of C'_q. □

Corollary 5.10. Given a pair of configurations C_p, C_q with empty sets of events, $C_p \sim C_q$, tr is a sequence of sets of external events. Then the sets of all possible response traces of C_p and C_q with respect to trace tr are the same.

Proof. Using the above theorem it is trivial. □

Corollary 5.11. Given two Statecharts P and Q, $P \sim Q$, tr is a sequence of sets of external events. Then the sets of all possible response traces of $D(P)$ and $D(Q)$ with respect to trace tr are the same.

6 Related work

The original Statecharts semantics is present by Harel et al. [6]. It obeys causality and synchrony, but not compositionality. The synchrony implies that the system is definitely faster than its environment, and can always finish computing its response before the next stimulus from the environment arrives. In 1991, A.Pnueli and M.Shalev [14] presented a way of defining the notion of step in the execution of Statecharts. This semantics maintains the synchrony hypothesis. They defined the function $En(\tau)$ and used it to describe the synchrony, causality and global consistency formally. They also gave a step-construction procedure to compute $En(\tau)$ for a Statechart with respect to a certain environment. In 1996, M. Schettini, A.Peron and S.Tini [16] gave a new definition which covered the definition in [14] and included a new restriction named compatibility, such that their step-construction procedure will not fail.

With regard to a semantics for Statecharts, it is very important whether it is compositional or not. Because the compositionality ensures that the semantics for a Statechart can be defined in terms of its component-charts. This is important especially when only a few components of a large Statechart change, a waste of resources by re-compiling the large Statechart will not take place. Theoretical studies constructed by Huizing [10] showed that one cannot combine the features of causality, synchrony hypothesis and compositionality with a step semantics which labels transitions by sets of "input/output" events. G. Lüttgen, M. von der Beeck and R. Cleaveland [12] presented an approach to define Statecharts' semantics. Their semantics achieved compositionality on the explicit micro-step level and causality and synchrony on the implicit macro-step

level. Our semantics is compositional. It adopts an asynchronous time model, in which a macro-step is defined as a sequence of micro-steps taking place instantaneously. To be more intuitive, our semantics obeys local consistency rather than global one. Furthermore, our semantics supports the data-states issues of Statecharts, i.e. the actions in a transition can contain assignments.

In [16], the equivalences of Statecharts are investigated. The authors associated a Labeled Transition System (LTS) with each Statechart term in a syntax directed way, and defined the semantics of Statecharts based on the LTS. They defined a causal order over events to express the causality. Using these notions, they defined four levels of equivalence of Statecharts and proved the properties of congruence respectively. The main difference between our work and what presented in [16] is as follows. The first definition of equivalence in [16] needs a bijection between all possible configurations of the two Statecharts whose equivalence is under consideration. It causes much troublesome in proving the property of congruence. Our concept of equivalence is similar to the second definition of equivalence of [16]. We only need the bisimulation of the default states of the Statecharts, which makes it much easier to prove the congruence property (**Theorem 4.11**). It seems that our definition is weaker in comparison to the first definition of equivalence in [16], but, in fact, it is not. As we have proved in **Theorem 4.9**, our concept has all the expected properties of equivalence stated in the first definition of equivalence in [16] and, at the same time, can get rid of the redundant statements in proving the properties of congruence for that level of equivalence, thus, getting the same results in a much simpler way.

C.A.R.Hoare [8] defined the trace notations of CSP. Many scholars defined the trace notations for other languages, for instance [9, 3, 13, 7], to describe observable behaviours of systems. Borrowing the ideas from these work we define the trace notations for Statecharts as sequences of sets of external stimuli and sequences of responses of the Statechart to these stimuli. Some properties with respect to the trace model for Statecharts are also explored. We believe these definitions can be valuable in further investigation of Statecharts' properties at behavioural level of traces.

7 Conclusions and future work

In this paper we have explored a set of transition rules so as to describe the operational semantics of Statecharts. We introduced the bisimulation to illustrate the equivalence between Statecharts' configurations. Ulteriorly we defined the equivalence between Statecharts and studied congruence properties with respect to the construction operators of Statecharts (*And* and *Or* constructions). In the end we introduced the notions of traces of Statecharts. It is foreseeable that we can describe the equivalence of Statecharts at the level of observable traces. As part of future work, the trace model should be further refined to comprise more information on behaviours of Statecharts, like causal orders of events generated in one instant, instantaneous updates of data state. The simulation

between Statecharts needs also to be investigated to describe the refinement of Statecharts.

Acknowledgement. We would like to thank anonymous referees for many helpful comments.

References

1. M. von der Beeck, A comparison of Statecharts variants, *Formal Tech. in Real-Time and Fault-Tolerant Systems*, LNCS 863, pp.128-148, Springer-Verlag, 1994.
2. G. Booch, J. Rumbaugh and I. Jacobson, *The Unified Modeling Language User Guide*, Addison-Wesley Longman, Reading, MA, USA, 1998.
3. J. Davies and S. Schneider. A brief history of Timed CSP. *Theoretical Computer Science*, 138:243–271, 1995.
4. D. Harel, Statecharts: a visual formalism for complex systems, *Science of Computer Programming*, 8(3), pp.231-274, 1987.
5. D. Harel and A. Naamad, The STATEMATE semantics of Statecharts, *ACM Trans. on Software Engineering and Methodology*, 5(4), pp.293-333, Oct. 1996.
6. D. Harel, A.Pnueli, J.Schmidt, and R.Sherman, On the formal semantics of Statecharts, *Symp. on Logis in Computer Science*, pp.56-64, IEEE CS Press, 1987.
7. J. He, An algebraic approach to the VERILOG programming, *Proc. of 10th Anniversary Colloquium of the United Nations University / International Institute for Software Technology*, Springer-Verlag, 2002.
8. C. A. R. Hoare, *Communicating Sequential Processes*, Prentice Hall, 1985.
9. C. A. R. Hoare and J. He, *Unifying Theory of Programming*, Prentice Hall, 1998.
10. C. Huizing, *Semantics of Reactive Systems: Comparison and Full Abstraction*, Ph.D. thesis, Eindhoven University of Technology, The Netherlands, 1991.
11. N. G. Leveson, M. Heimdahl, H. Hildreth, and J. Reese, Requirements specifications for process control systems, *IEEE Trans. on Software Engineering*, 20(9), pp.684-707, Sept. 1994.
12. G. Lüttgen, M. von der Beeck, and R. Cleaveland, A compositional approach to Statecharts semantics, NASA/ICASE Report No.2000-12, March 2000.
13. B. Mahony and J. S. Dong. Overview of the semantics of TCOZ. *IFM'99: Integrated Formal Methods*, pp. 66–85. Springer-Verlag, 1999.
14. A. Pnueli and M. Shalev, What is in a step: on the semantics of Statecharts, *Theo. Aspects of Computer Software*, LNCS 526, pp.244-264, Springer-Verlag, 1991.
15. S. Qin and W. N. Chin, Mapping Statecharts to VERILOG for hardware/software co-specification, *FM03: the 12th International FME Symposium (to appear)*, 2003.
16. M. Schettini, A. Peron, and S. Tini, Equivalences of Statecharts, *7th International Conference on Concurrency*, LNCS 1119, pp.687-702, Springer-Verlag, Aug. 1996.